# Eliminating Trapping Sets in Low-Density Parity Check Codes by using Tanner Graph Covers

Miloš Ivković, Shashi Kiran Chilappagari
and Bane Vasić, *Fellow, IEEE*

***Abstract*— We discuss error floor asympotics and present a method for improving the performance of low-density parity check (LDPC) codes in the high SNR (error floor) region. The method is based on Tanner graph covers that do not have trapping sets from the original code. The advantages of the method are that it is universal, as it can be applied to any LDPC code/channel/decoding algorithm and it improves performance at the expense of increasing the code length, without losing the code regularity, without changing the decoding algorithm, and, under certain conditions, without lowering the code rate. The proposed method can be modified to construct convolutional LDPC codes also. The method is illustrated by modifying Tanner, MacKay and Margulis codes to improve performance on the binary symmetric channel (BSC) under the Gallager B decoding algorithm. Decoding results on AWGN channel are also presented to illustrate that optimizing codes for one channel/decoding algorithm can lead to performance improvement on other channels.**

***Index Terms*— convolutional LDPC codes, error floor, Gallager B, LDPC codes, min-sum decoding algorithm, Tanner code, trapping sets.**

## I. Introduction

The error-floor problem is arguably the most important problem in the theory of low-density parity check (LDPC) codes and iterative decoding algorithms. Roughly, error floor is an abrupt change in the frame error rate (FER) performance of an iterative decoder in the high signal-to-noise ratio (SNR) region (see [9] for more details and [1], [2], [3] for general theory of LDPC codes).

The error floor problem for iterative decoding on the binary erasure channel (BEC) is now well understood, see [7], [8] and the references therein.

In the case of the additive white Gaussian noise (AWGN) channel, MacKay and Postol in [4] pointed out a weakness in the construction of the Margulis code [22] which led to high error floors. Richardson [9] presented a method to estimate error floors of LDPC codes and presented results on the AWGN channel. He pointed out that the decoder performance is governed by a small number of likely error events related to certain topological structures in the Tanner graph of the code, called *trapping sets* (or *stopping sets* on BEC [7]).[1] The approach from [9] was further refined by Stepanov *et al.* in [10]. Zhang *et al.* [11] presented similar results based on hardware decoder implementation. Vontobel and Koetter [12] established a theoretical framework for finite length analysis of message passing iterative decoding based on graph covers. This approach was used by Smarandache *et al.* in [13] to analyze the performance of LDPC codes from projective geometries [13] and for LDPC convolutional codes [14].

An early account on the most likely error events on the binary symmetric channel (BSC) for codes which Tanner graphs have cycles is given by Forney *et al.* in [16]. Some results on LDPC codes over the BSC appear in [13], as well.

M. Ivković is with the Department of Mathematics, University of Arizona Tucson, AZ 85721, USA, e-mail: milos@math.arizona.edu.

S. K. Chilappagari and B. Vasić are with Dept. of Electrical and Computer Eng. Univ. of Arizona.

An earlier version of this work was presented at the 2007 IEEE Intern. Symposium on Information Theory (ISIT'07).

This work was supported by grants from INSIC-EHDR and NSF-CCR (Grant no. 0634969).

[1]The necessary definitions will be given in the next section.

A significant part of the research on error floor analysis has also focused on methods for lowering the error floor. The two distinct approaches taken to tackle this problem are (1) modifying the decoding algorithm and (2) constructing codes avoiding certain topological structures. Numerous modifications of the sum-product decoding algorithm were proposed, see, for example, [18] and [19], among others.

Among the methods from the second group, there have been novel constructions of codes with high Tanner graph girth [21], [6], as it was observed that codes with low girth tend to have high error floors. While it is true that known trapping sets have short cycles [10], [17], the example of projective geometry codes, that have short cycles, but perform well under (hard decision) iterative decoding, suggests that maximizing the girth is not the optimal procedure. As the understanding of the error floor phenomena and its connection with trapping sets grows, avoiding the trapping sets directly (rather than short cycles) seems to be a more efficient way (in terms of code rate and decoding complexity), to suppress error floors.

Code modification for improving the performance on the binary erasure channel (BEC) was studied by Wang in [20]. To the best of our knowledge, it is the first paper on code modification with maximizing the size of stopping (or trapping) sets as the objective. Edge swapping within the code was suggested as a way to break the stopping sets. The method that we propose is similar. Roughly speaking, it consists of taking two (or more) copies of the same code and swapping edges between the code copies in such a way that the most dominant trapping sets are broken. It is also similar to the code constructions that appear in Smarandache *et al.* [14], Thorpe [24], Divsalar and Jones [25] and Kelley, Sridhara and Rosenthal [26].

The advantages of the method are: (a) it is universal as it can be applied to any code/channel model/decoding algorithm and (b) it improves performance at the expense of increasing the code length only, without losing the code regularity, without changing the decoding algorithm, and, under certain conditions, without lowering the code rate. If the length of the code is fixed to $n$, the method can be applied by taking $t$ copies of a (good) code $C$ of length $n/t$ and eliminating the most dominant trapping sets of $C$. The method can be slightly modified to construct convolutional LDPC codes as well. The details are given in Section III.

We apply our method and construct codes based on Margulis [22], Tanner [21] and MacKay [23] codes and present results on the BSC when decoded using the Gallager B algorithm [1]. It is worth noting that the error floor on the AWGN channel depends not only on the structure of the code but also on implementation nuances of the decoding algorithm, such as numerical precision of messages [9]. Since the Gallager B algorithm operates by passing binary messages along the edges of a graph, any concern about the numerical precision of messages does not arise.

The rest of the paper is organized as follows. In Section II we introduce the notion of trapping sets and their relation to the performance of the code. We explain the proposed method in Section III. We present numerical results in Section IV and conclude in Section V.

## II. Basic Concepts

The Tanner graph of an LDPC code, $\mathcal{G}$, is a bipartite graph with two sets of nodes: variable (bit) nodes and check (constraint) nodes. The nodes connected to a certain node are referred to as its neighbors. The degree of a node is the number of its neighbors. The girth $g$ is the length of the shortest cycle in $\mathcal{G}$. In this paper, ● represents a variable node, □ represents an even degree check node and ■ represents an odd degree check node.

The notion of trapping sets was first introduced in [4], but here we follow the formalism from [19].

*Definition 1:* For a given $m \times n$ matrix $U = (U_{i,j})$ with $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, *the projection* of a set of $h$ columns indexed by $j_1, j_2, \ldots, j_h$ is an $m \times h$ matrix consisting of the elements $u_{i,j}$, $1 \leqslant i \leqslant m$, $j = j_1, j_2, \ldots, j_h$.

*Definition 2:* Let $H$ be a parity check matrix of an LDPC code. An $(a, b)$ *trapping set* $\mathfrak{T}$ is a set of $a$ columns of $H$ with a projection that contains $b > 0$ odd weight rows.

The definition of the trapping set above is purely topological, that is, a trapping set can be seen as a subgraph of the Tanner graph. In other words, an $(a, b)$ trapping set $\mathcal{T}$ is a subgraph with $a$ variable nodes and $b$ odd degree checks. The most probable noise realizations that lead to decoding failure are related to trapping sets ([9], [10]). A measure of noise realization probability is referred to as *pseudo-weight.* Following the terminology in [10], an *instanton* can be defined as the most likely noise realization that leads to decoding failure.

The instantons on the BSC consist of the received bit configurations with minimal number of erroneous bits that lead to decoding failure. Following [17], the notion specific to BSC, analogous to pseudo-weight, can be defined as:

*Definition 3:* The minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set $\mathfrak{T}$ will be referred to as *the critical number* $k$ for that trapping set.

*Remark:* To "end up" in a trapping set $\mathfrak{T}$ means that, after a finite number of iterations, the decoder will be in error, on at least one variable node from $\mathfrak{T}$, at every iteration. Note that the variable nodes that are initially in error do not have to be within the trapping set.

We illustrate the above concepts with an example.

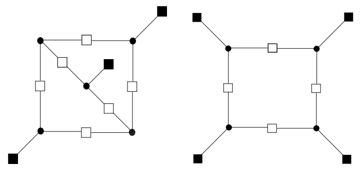

(a) (5,3) trapping set (b) (4,4) trapping set

Fig. 1. Trapping sets

*Example 1:* The $(5, 3)$ trapping set in Fig. 1(a). appears (among other codes) in the Tanner (155, 64) code [17] (see also the examples of *irreducible closed walks* in the chapter 6.1 of [5]) . This trapping set has critical number $k = 3$ under the Gallager B decoding algorithm (for the definition of the algorithm see [2]), meaning that, if three variable nodes, on the diagonal from bottom left to top right, are initially in error, the decoder will fail to correct the errors.

Fig. 1(b) illustrates a $(4, 4)$ trapping set. This trapping set, although smaller, has critical number $k = 4$, (all the variable nodes have to be in error initially for the decoder to fail). So, if a code has both $(5, 3)$ and $(4, 4)$ trapping sets, the FER performance is dominated by the $(5, 3)$ trapping set.

At the end of this example, we note that the $(5, 3)$ trapping set above is an example of an *oscillatory trapping set,* i.e, if three variable nodes on the diagonal are initially in error, after the first iteration those three nodes will be decoded correctly, but the remaining two will be in error. In the decoding attempt after the second iteration those two will be correct, but the initial three will be in error again, and so on.

*Remark:* Note that on the BEC the critical number is just the size of the stopping set, see [20].

We now clarify what "the most dominant trapping sets" means and how these effect code performance.

Let $\alpha$ be the transition probability of the BSC and $c_k$ be the number of configurations of received bits for which $k$ channel errors lead to a codeword (frame) error. The frame error rate (FER) is given by:

$$FER(\alpha) = \sum_{k=i}^{n} c_k \alpha^k (1-\alpha)^{(n-k)}$$

where $i$ is the minimal number of channel errors that can lead to a decoding error (size of instantons) and $n$ is the length of the code.

On a semilog scale the FER is given by the expression

$$\log(FER(\alpha)) = \log\big(\sum_{k=i}^{n} c_k \alpha^k (1-\alpha)^{n-k}\big) \quad (1)$$

$$= \log(c_i) + i\log(\alpha) + \log((1-\alpha)^{n-i}) \quad (2)$$

$$+\log\left(1 + \frac{c_{i+1}}{c_i}\alpha(1-\alpha)^{-1} + \ldots + \frac{c_n}{c_i}\alpha^{n-i}(1-\alpha)^{i-n}\right) \quad (3)$$

In the limit $\alpha \to 0$ we note that

$$\lim_{\alpha\to 0}\Big[\log((1-\alpha)^{n-i})\Big] = 0$$

and

$$\lim_{\alpha\to 0}\Big[\log\Big(1 + \frac{c_{i+1}}{c_i}\alpha(1-\alpha)^{-1}\ldots + \frac{c_n}{c_i}\alpha^{n-i}(1-\alpha)^{i-n}\Big)\Big] = 0$$

So, the behavior of the FER curve for small $\alpha$ is dominated by

$$\log(FER(\alpha)) \approx \log(c_i) + i\log(\alpha)$$

The $\log(FER)$ vs $\log(\alpha)$ graph is close to a straight line with slope equal to $i$ -the minimal critical number or cardinality of the instantons.

Therefore, if two codes $C_1$ and $C_2$ have instanton sizes $i_1$ and $i_2$, such that $i_1 < i_2$, then the code $C_2$ will perform better than $C_1$ for small enough $\alpha$, independent of the number of instantons, just because $\log(\alpha) \to -\infty$ as $\alpha \to 0$. Note also that the critical number of the most dominant trapping sets cannot be greater than half the minimum distance. If it is the case, the performance of the decoder is dominated by the minimum weight codewords.

## III. The Method for Eliminating Trapping Sets

In this section we present a method to construct an LDPC code $C^{(2)}$ of length $2n$ from a given code $C$ of length $n$ and discuss a modification of the method that gives a convolutional LDPC code based on $C$.

Let $H$ and $H^{(2)}$ represent the parity check matrices of $C$ and $C^{(2)}$ respectively. $H^{(2)}$ is initialized to

$$H^{(2)} = \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix}.$$

Stated simply, $H^{(2)}$ is formed by taking two copies of $H$ say $C_1$ and $C_2$. It can be seen that if $H$ has dimensions $m \times n$, then $H^{(2)}$ has dimensions $2m \times 2n$. Every edge $e$ in the Tanner graph $\mathcal{G}$ of $C$ is associated with a nonzero entry $H_{t,k}$. The operation of changing the value of $H^{(2)}_{t,k}$ and $H^{(2)}_{m+t,n+k}$ to "0", and $H^{(2)}_{m+t,k}$ and $H^{(2)}_{m,n+k}$ to "1" is termed as swapping the edge $e$. Fig. 2 illustrates edge swapping in two copies of a $(5, 3)$ trapping set. We assume that the most

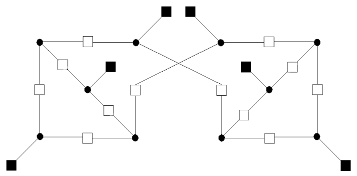


Fig. 2. Trapping set elimination

dominant trapping sets for $C$ are known. The method can be described in the following steps.

**Algorithm:**

1) Take two copies $C_1$ and $C_2$ of the same code. Since the codes are identical they have the same trapping sets. Initialize *SwappedEdges*=$\phi$; *FrozenEdges*=$\phi$;
2) Order the trapping sets by their critical numbers.
3) Choose a trapping set $\mathfrak{T}_1$ in the Tanner graph of $C_1$, with minimal critical number. Let $E_{\mathfrak{T}_1}$ denote the set of all edges in $\mathfrak{T}_1$. If ($E_{\mathfrak{T}_1} \cap$ *SwappedEdges* $\neq \phi$) goto 5. Else goto 4.
4) Swap an arbitrarily chosen edge $e \in E_{\mathfrak{T}_1} \setminus$ *FrozenEdges* (if it exists). Set *SwappedEdges* = *SwappedEdges* $\cup$ $e$.
5) "Freeze" the edges $E_{\mathfrak{T}_1}$ from $\mathfrak{T}_1$ so that they cannot be swapped in the following steps. Set *FrozenEdges* = *FrozenEdges* $\cup E_{\mathfrak{T}_1}$.
6) Repeat steps 2 to 4 until it is possible to remove the trapping sets of the desired size.

Step 5 is needed because swapping additional edges from the (former) trapping sets might introduce trapping sets with a same critical number again. Fig. 3 illustrates such a swapping which corresponds to just interchanging the check nodes.

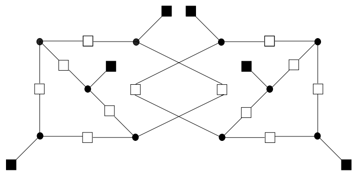


Fig. 3. Reintroducing trapping set by swapping two edges

The Tanner graph of the newly made code is a special double cover of the original code's Tanner graph, interested readers are referred to [12].

*Remark:* There are several approaches which may improve the efficiency of the algorithm. Firstly, instead of swapping the edges at random at step 3, edges could be swapped based on the number of trapping sets they participate in, or by using some other schedule that would (potentially) lead to the highest number of trapping sets eliminated. The structure of the code can also be exploited. For example, the Margulis $(2640, 1320)$ code [22], has $1320$ $(4, 4)$ minimal trapping sets with the property that each trapping set has one edge that does not participate in any other minimal trapping set. So, instead of swapping edges at random, the edges appearing in only one trapping set can be swapped, and such a procedure is guaranteed to eliminate all the minimal trapping sets. Also, there is a possibility not to freeze all the edges from the (former) trapping sets, but only those that would, if swapped, introduce the trapping sets with the same critical number.

Note, however, that any edge swapping schedule can be seen as a particular realization of the random edge swapping. For all the codes that we considered, all trapping sets with minimal critical number were eliminated by the algorithm with random edge swapping.

The following theorem shows how this method affects the code rate.

*Theorem 1:* If the code $C$, with parity check matrix $H$, and rate $r$ (and length $n$) is used in the algorithm above, the resulting code $C^{(2)}$ will have rate $r^{(2)}$ (and length $2n$), such that $r^{(2)} \leqslant r$.

*Proof:* Each edge swapping operation in the algorithm can be seen as matrix modification. At the end of the algorithm, code $C^{(2)}$ is determined by

$$H^{(2)} = \begin{bmatrix} H' & B \\ B & H' \end{bmatrix}$$

where $H'$ and $B$ are matrices such that $H' + B = H$, and $H'_{t,k}$ (or $B_{t,k}$) can be equal to "1" only if $H_{t,k} = 1$.

If the second block row is added to the first in $H^{(2)}$, and then the the first block column is added to the second, we end up with

$$\begin{bmatrix} H' & B \\ B & H' \end{bmatrix} \to \begin{bmatrix} H & H \\ B & H' \end{bmatrix} \to \begin{bmatrix} H & 0 \\ B & H \end{bmatrix} \tag{4}$$

The last matrix in (4) has rank which is greater than or equal to twice the rank of $H$. Therefore, the code $C^{(2)}$ has rate $r^{(2)} \leqslant r$ where $r$ is the rate of $C$.□

Note, that $r^{(2)} = r$ if $B = CH + HD$, for some matrices $C$ and $D$, so that $CH$ corresponds to linear combinations of rows of $H$ and $HD$ corresponds to linear combinations of columns of $H$. We also have a following corollary.

*Corollary 1:* If the matrix $H$ has full rank, then $r^{(2)} = r$.

*Proof:* This follows from the fact that if $H$ has full rank, then the last matrix in (4) has full rank also. □

At the end of this section, we briefly discuss the minimal distance of the modified code.

*Theorem 2:* If the code $C$ has minimal distance $d_{min}$, the modified code $C^{(2)}$, will have the minimal distance $d_{min}^{(2)}$, such that, $2d_{min} \geq d_{min}^{(2)} \geq d_{min}$.

*Proof:* We first prove that $d_{min}^{(2)} \geq d_{min}$. Suppose that the minimal weight codeword of $C^{(2)}$ is $c^{(2)}$, where $c^{(2)}$ is a column vector consisting of two vectors $c_1$ and $c_2$ of length $n$. Then $H^{(2)}c^{(2)} = 0$ is equivalent to

$$\begin{bmatrix} H' & B \\ B & H' \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} H'c_1 + Bc_2 \\ Bc_1 + H'c_2 \end{bmatrix} = 0 \tag{5}$$

Note that $c_1 + c_2 = c$ is a column vector of length $n$, with Hamming weight $w_h(c) \leq w_h\left(c^{(2)}\right)$, where $w_h\left(c^{(2)}\right)$ is the Hamming weight of the $c^{(2)}$. Now:

$$Hc = (H'+B)(c_1+c_2) = H'c_1 + Bc_1 + H'c_2 + Bc_2 = 0 \tag{6}$$

because the last expression in Eq. (6) is equal to the sum of entries of the last column vector in Eq. 5. So, $c$ is a codeword of $C$.

If $c \neq 0$, from $w_h(c) \leq w_h\left(c^{(2)}\right)$ it follows that $d_{min}^{(2)} \geq d_{min}$. If $c = 0$ then $c_1 = c_2$, and from Eq. (5) follows that $Hc_1 = 0$, so $c_1$ is a codeword of $C$ and again $d_{min}^{(2)} \geq d_{min}$.

The proof that $2d_{min} \geq d_{min}^{(2)}$ is similar. If we assume that $c_1$ is a minimal weight codeword of $C$, we have:

$$\begin{bmatrix} H' & B \\ B & H' \end{bmatrix} \begin{bmatrix} c_1 \\ c_1 \end{bmatrix} = 0 \tag{7}$$

so $2d_{min} \geq d_{min}^{(2)}$.

We finish this proof by mentioning that it is not difficult to construct examples where $2d_{min} = d_{min}^{(2)}$ or $d_{min}^{(2)} = d_{min}$, so the statement of the theorem is "sharp". □

We described the algorithm in its basic form. $H^{(2)}$ can be initialized by interleaving the copies $C_1$ and $C_2$ in an arbitrary order, but we choose concatenation to keep the notation simple. The method, as well as all the proofs, will hold for any interleaving. It is also possible to consider more than two copies of the code to further eliminate trapping sets with higher critical number.

The splitting of parity check matrix $H$ into $H'$ and $B$ can be seen as a way to construct convolutional LDPC codes, that is, as a way to *unwrap* the original LDPC code $C$. For details on unwrapping see [15] and the references therein. The (infinite) parity check matrix can be can be constructed as:

$$H_{conv} = \begin{bmatrix} H' & & & & \\ B & H' & & & \\ & B & H' & & \\ & & B & \ddots & \\ & & & \ddots & \end{bmatrix} \quad (8)$$

Note that by construction the resulting convolutional code has pseudo-codewords with higher pseudo-weights than original LDPC code. In this light, Theorem 2 can be seen as a generalization of Lemma 2.4 from [14]. We refer readers interested in convolutional LDPC codes to that paper.

## IV. Numerical Results

In this section we illustrate the proposed method by modifying the Margulis [22], Tanner [21] and MacKay [23] codes to eliminate trapping sets under the Gallager B decoding algorithm. We use the trapping sets reported in [17].

*Example 2: (Margulis* $(2640, 1320)$ *code)* The parity check of this matrix has full rank, so the modified code is an $(5280, 2640)$ code, and has the same rate as the original code, i.e., $r^{(2)} = r = 0.5$.

This code has 1320 $(4,4)$ trapping sets with critical number 4 as the most dominant ones. The modified $(5280, 2640)$ code has no $(4,4)$ trapping sets and the performance is governed by $(5,5)$ trapping sets (ten cycles), that have critical number $k = 5$, Fig. 4.

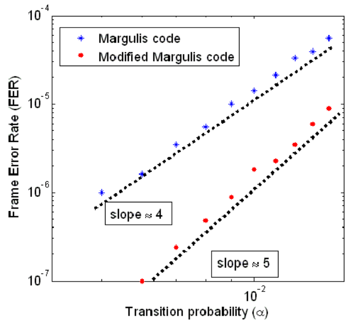


Fig. 4. Margulis code performance

*Example 3: (Tanner (155, 64) code)* This code has $(5,3)$ trapping sets (Fig. 1(a)) with critical number $i = 3$ as the most dominant ones. There are 155 such trapping sets [17], [21]. In this case we used a version of the method in which it is possible to swap edges from the (former) trapping sets, if no trapping set of the same or smaller critical number is introduced. The result was a (310, 126) code for which the minimal trapping sets are type (4,4) (eight cycles) with critical number $k = 4$ (see Fig 1(b)). This was confirmed by numerical simulations in Fig. 5. The FER curve changes the slope, for higher $\alpha$, where FER contribution from the expression (3) is not negligible. Note that there was a small rate penalty to this procedure. The original Tanner code has rate 0.4129, whereas the modified code has rate 0.4065.

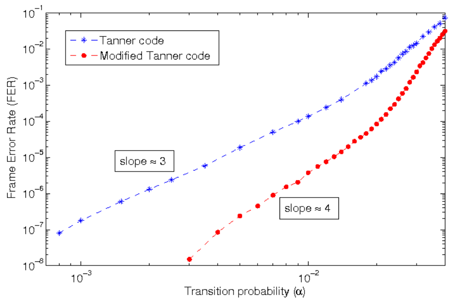


Fig. 5. Tanner code performance for a longer range of $\alpha$

*Example 4: (MacKay's (1008, 504) codes)* This is an example of how the method can be used to produce better codes of a fixed length. We have taken a 504 length MacKay code and constructed a 1008 $(2 * 504)$ length code. The new code performs better than MacKay codes of length 1008.

Both original 504 and 1008 length codes have two types of trapping sets with critical number $k = 3$, (5,3) and (3,3) (six cycles). We ran the algorithm so that all (3,3) trapping sets are eliminated from the newly constructed, but none of the (5,3) trapping sets. The results are shown in Fig. 6. It can be seen that, although the FER performance

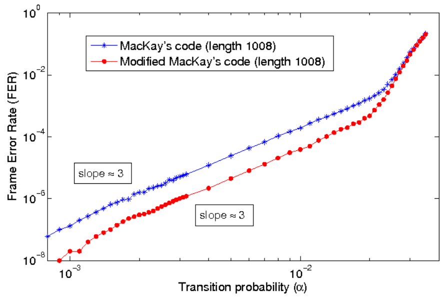


Fig. 6. MacKay's codes performance

is improved, the slope of the FER curve is approximately the same.[2]

*Example 5: (AWGN channel)* This example illustrates two points. First is that optimizing code for one decoding algorithm can lead to performance improvement for other decoding algorithms. The second point is that the use of an appropriate axis scaling can greatly help in error floor analysis and code performance prediction.

We present FER results over AWGN channel and min-sum algorithm after 500 iterations for three codes, the original Tanner (155, 64) code, our modified Tanner (310, 126) from the Example 3 and a random (310, 127) code with column weight 3 and row weight 5.

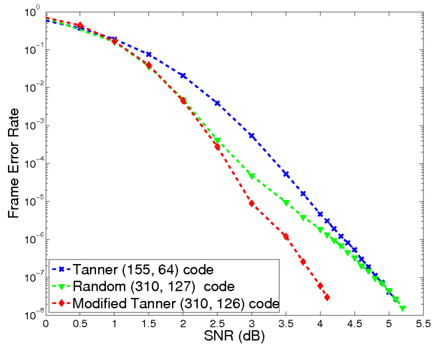
In the low SNR region, where all kinds of error events are likely, the length (and rate) of a code govern the performance. In this region codes of length 310 have similar performance. For high SNRs, however, code optimization in terms of trapping sets becomes important and random code performance becomes much worse than performance of the modified Tanner (310, 126) code. Notice a pronounced error floor for the random code.

What is even more illustrative is Fig. 7(b) where we plot $\log(FER)$ versus SNR (not in dB) on the x-axis. This is because for high SNRs on the AWGN channel, similarly to Eq. (3), $FER \propto \exp(-\omega_{in} * \text{SNR}/2)$, where $d_{in}$ is pseudo-weight of the most likely

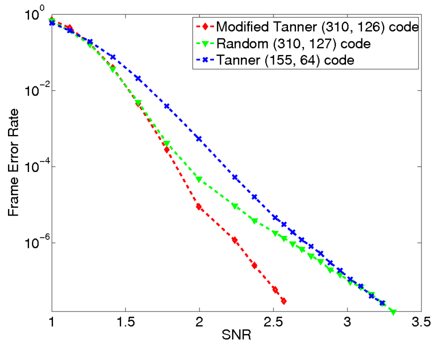
[2]It is possible that a more sophisticated algorithm would also eliminate the (5,3) trapping sets. However, our goal with this example was to show the performance when some, but not all, of the trapping sets with minimal critical number are eliminated.

error event. So on the graph with SNR on the x-axis which is not in dB, $\log(FER)$ curve will approach (from above) a straight line with slope equal to $-\omega_{in}/2$ as SNR $\to \infty$. See [5] and [12] for further details. Using these observations and numerical results obtained by simulations we can estimate that our modified code has the slope approximately equal to 20, better than the original Tanner (155, 64) code with the slope of $\approx 14$.[3]

Further more, considering that the slope for the random code is $\approx 12$, we can claim that, for SNR values higher than those on the plots, the Tanner code will perform better than the random code.



(a) $\log(FER)$ versus SNR in dB



(b) $\log(FER)$ versus SNR as $\frac{E}{N}$ (not in dB)

Fig. 7. FER performance under min-sum decoding

## V. Conclusion

The proposed method allows the construction of codes with good FER performance, but low row/column weight (as opposed to projective geometry codes) and therefore relatively low decoding complexity. Although numerical results for the Gallager B decoder are presented, we reiterate that the method can be used for code optimization based on the trapping sets of an arbitrary decoder.

The algorithm can also be used to determine the pseudo-weight spectrum of a code as follows. Once the most likely trapping sets (those with the smallest pseudo-weight) are determined and eliminated by the method, the numerically obtained decoding performance of a modified code, i.e., the slope of the FER curve with appropriate axis, gives an estimate of the pseudo-weight of the next most likely trapping sets -just as it was done in the Example 5 with the Tanner code and the modified Tanner code.

## Acknowledgment

The authors would like to acknowledge valuable discussions with Robert Indik, Misha Stepanov and Clifton Williamson.

[3]The estimate for the Tanner code is in accordance with the pseudo-weight of the single most likely error event of $\approx 12.45$ reported in [10].

## References

[1] R. G. Gallager, *Low Density Parity Check Codes*, Cambridge, MA: MIT Press, 1963.
[2] T. J. Richardson and R. Urbanke, “The Capacity of Low-Density Parity-Check Codes Under Message-Passing Algorithm,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599-618, Feb. 2001.
[3] T. Richardson, A. Shokrollahi, and R. Urbanke, ”Design of capacity approaching irregular low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, 2001.
[4] D. MacKay and M. Postol, “Weaknesses of Margulis and Ramanujan-Margulis low-density parity check codes,” Electronic Notes in Theoretical Computer Science, vol. 74, 2003.
[5] N. Wiberg, *Codes and decoding on general graphs*, Ph.D. thesis, Linköping University, 1996.
[6] J.K. Moura. J. Lu, and H. Zhang, “Structured LDPC codes with large girth,” IEEE Signal Proc. Mag., vol. 21, no. 1, pp. 42-55, Jan. 2004
[7] C. Di *et al.*, “Finite length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Trans. on Information Theory*, vol. 48, no. 6, pp. 1570-1579, Jun 2002.
[8] C. Wang, S. R. Kulkarni, H. V. Poor, “Upper Bounding the Performance of Arbitrary Finite LDPC Codes on Binary Erasure Channels,” in *Proc. Intern. Symp. on Inform. Theory*, Seattle, WA, USA, July 9-14, 2006.
[9] T. J. Richardson, “Error Floors of LDPC Codes,” in Proc. 41st Annual Allerton Conf. on Communications, Control and Computing, 2003.
[10] M. Stepanov, M. Chertkov, “Instanton analysis of Low-Density Parity-Check codes in the error-floor regime,” , *Proc. IEEE Intern. Symp. on Inform. Theory (ISIT)*, Seattle, WA, USA, July 9-14, 2006.
[11] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam and M Wainwright “Investigation of Error Floors of Structured Low- Density Parity-Check Codes by Hardware Emulation, in *Proc. IEEE Globecom*, San Francisco, CA, USA, 2006.
[12] P. O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” preprint, available online at http://www.arxiv.org/.
[13] R. Smarandache and P. O. Vontobel,“Pseudo-Codeword Analysis of Tanner Graphs from Projective and Euclidean Planes,” preprint, available online at http://arxiv.org/.
[14] R. Smarandache, A. E. Pusane, P. O. Vontobel and D.J Costello, Jr.,“Pseudo-Codeword Performance Analysis for LDPC Convolutional Codes,” preprint, available online at http://arxiv.org/.
[15] A. E. Pusane, R. Smarandache, P. O. Vontobel and D.J Costello, Jr., “On deriving good LDPC convolutional codes from OC LDPC Block Codes,” in *Proc. IEEE Intern. Symp on Inform. Theory*, Nice, France, June 24-29, 2007, pp. 1221-1225.
[16] G. D. Forney, Jr., R. Koetter, F. R. Kschischang, and A. Reznik, “On the effective weights of pseudocodewords for codes defined on graphs with cycles,” in Codes, Systems, and Graphical Models (Minneapolis, MN, 1999) (B. Marcus and J. Rosenthal, eds.), vol. 123 of IMA Vol. Math. Appl., pp. 101–112, Springer Verlag, New York, Inc., 2001.
[17] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, “Error Floors of LDPC Codes on the Binary Symmetric Channel,” *Proc. Intern. Conf. on Communications*, ICC 2006, Istanbul, Turkey, June 2006.
[18] N. Varnica and M. Fossorier, “Improvements in belief-propagation decoding based on averaging information from decoder and correction of clusters of nodes,” in *IEEE Communications Letters*, vol. 10, no 12. pp 846-848, Dec. 2006.
[19] S. Laendner, T. Hehn, O. Milenković and J. Huber, “When does one redundant parity-check equation matter?” in *Proc. IEEE Globecom*, San Francisco, CA, USA, 2006.
[20] C. Wang, “Code annealing and the suppressing effect of the cyclically lifted LDPC code ensemble,” presented at *2006 IEEE Information Theory Workshop*, Chengdu, China, October 22-26, 2006.
[21] R. M. Tanner, D. Sridhara and T. Fuja, “A class of Group-Structured LDPC codes,” in *Proc. ISCTA*, 2001.
[22] J. Rosenthal and P. O. Vontobel, “Constructions of LDPC Codes Using Ramanujan Graphs and Ideas From Margulis,” in *Proc. of the 38-th Allerton Conf. on Communication, Control, and Computing*, 2000.
[23] D. J. C. MacKay, “Encyclopedia of Sparse Graph Codes,” Online: http://www.interference.phy.cam.ac.uk/ mackay/codes/data.html.
[24] J. Thorpe, “Low Density Parity Check (LDPC) Codes Constructed from Protographs,” JPL INP Progress Report 42-154, August 15, 2003.
[25] D. Divsalar, C. Jones, “Protgraph LDPC Codes with Node Degrees at Least 3,” in *Proc. IEEE Globecom*, San Francisco, CA, USA, 2006.
[26] C. Kelley, D. Sridhara, J. Rosenthal,“Tree-Based Construction of LDPC Codes Having Good Pseudocodeword Weights,” preprint, available online at http://arxiv.org/.